

REPORT
FOR
EXTENDED
LAB1

```
1 .global reset
2 reset:
3     ldr sp, =stack_top
4     bl main
5 stop:
6     bl stop

MINGW32/e/Master/Embedded-System-Online-Diploma/Unit3_EmbeddedC/Unit3_EmbeddedC_lecture 3/Extended for lab1
qcap.dll qedit.dll quickassist.exe
Mohamed@DESKTOP-A7UUKKD MINGW32 /e/Master/Embedded-System-Online-Diploma/Unit3_E
mbeddedC/Unit3_EmbeddedC_lecture 3/Extended for lab1 (main)
$ qu
query.dll quartz.dll quickassist.exe
Mohamed@DESKTOP-A7UUKKD MINGW32 /e/Master/Embedded-System-Online-Diploma/Unit3_E
mbeddedC/Unit3_EmbeddedC_lecture 3/Extended for lab1 (main)
$ qed
qedit.dll qedwipes.dll
Mohamed@DESKTOP-A7UUKKD MINGW32 /e/Master/Embedded-System-Online-Diploma/Unit3_E
mbeddedC/Unit3_EmbeddedC_lecture 3/Extended for lab1 (main)
$ E:/Master/qemu/qemu-system-arm -M versatilepb -m 128M -nographic -kernel learn
Learn-in-Depth: Mohamed Abdallah

Mohamed@DESKTOP-A7UUKKD MINGW32 /e/Master/Embedded-System-Online-Diploma/Unit3_E
mbeddedC/Unit3_EmbeddedC_lecture 3/Extended for lab1 (main)
$ E:/Master/qemu/qemu-system-arm -M versatilepb -s -S -m 128M -nographic -kernel learn-in-depth.elf |

9 .debug_loc 00000058 00000000 00000000 00008548 2**0
10 .debug_str 0000011d 00000000 00000000 000085a0 2**0
11 .debug_frame 00000054 00000000 00000000 000086c0 2**2
CONTENTS, READONLY, DEBUGGING
CONTENTS, READONLY, DEBUGGING

Mohamed@DESKTOP-A7UUKKD MINGW32 /e/Master/Embedded-System-Online-Diploma/Unit3_EmbeddedC/Unit3_EmbeddedC_lecture 3/Extended for lab1 (main)
$ arm-none-eabi-gdb.exe learn-in-depth.elf
GNU gdb (GDB) 7.5.1
Copyright (C) 2012 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software; you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law. Type "show copying"
and "show warranty" for details.
This GDB was configured as "--host=i686-pc-mingw32 --target=arm-none-eabi".
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>...
Reading symbols from E:/Master/Embedded-System-Online-Diploma/Unit3_EmbeddedC/Unit3_EmbeddedC_lecture 3/Extended for lab1/learn-in-depth.elf
(gdb) Target remote localhost:1234
Remote debugging using localhost:1234
reset () at startup.s:3
3     ldr sp, =stack_top
(gdb) |
```

1. Startup.s file (assembly).
2. Turning on our board (versatilepb) and burning our barmetal software application (writing on uart Learn-in-Depth: Mohamed Abdallah).
3. My board is ready and wait debugger.
4. Path learn-in-depth.elf to debug (not .bin as it's not contain debug info).
5. Open path with my board (localhost → my machine IP, 1234 → port, info from board datasheet).
6. Processor @ entry point (reset section at startup file loading value of stack_top to stack pointer "initializing stack").

```
MINGW32/e/Master/Embedded-System-Online-Diploma/Unit3_EmbeddedC/Unit3_EmbeddedC_lecture 3/Extended fo...
$ arm-none-eabi-gdb.exe learn-in-depth.elf
GNU gdb (GDB) 7.5.1
Copyright (C) 2012 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law. Type "show copying"
and "show warranty" for details.
This GDB was configured as "--host=i686-pc-mingw32 --target=arm-none-eabi".
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>...
Reading symbols from E:\Master\Embedded-System-Online-Diploma\Unit3_EmbeddedC\Un
it3_EmbeddedC_lecture 3\Extended for lab1\learn-in-depth.elf...done.
(gdb) target remote localhost:1234
Remote debugging using localhost:1234
reset () at startup.s:3
3       ldr sp, =stack_top
(gdb) l
1       .global reset
2       reset:
3       ldr sp, =stack_top
4       bl main
5       stop:
6       bl stop(gdb)
(gdb) Line number 7 out of range; startup.s has 6 lines.
display/3i $pc
1: x/3i $pc
=> 0x10000 <reset>:  ldr    sp, [pc, #4]    ; 0x1000c <stop+4>
0x10004 <reset+4>:  bl     0x10010 <main>
0x10008 <stop>:    bl     0x10008 <stop>
(gdb) |
```

1. (1) show currently file.
2. (display/3i \$pc) show three instructions, currently program counter point to next instruction should be fetched (0x10000 initializing stack then fetch next instruction which will branch to main).

```
MINGW32/e/Master/Embedded-System-Online-Diploma/Unit3_EmbeddedC/Unit3_EmbeddedC_lecture 3/Extended fo...
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law. Type "show copying"
and "show warranty" for details.
This GDB was configured as "--host=i686-pc-mingw32 --target=arm-none-eabi".
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>...
Reading symbols from E:\Master\Embedded-System-Online-Diploma\Unit3_EmbeddedC\Un
it3_EmbeddedC_lecture 3\Extended for lab1\learn-in-depth.elf...done.
(gdb) target remote localhost:1234
Remote debugging using localhost:1234
reset () at startup.s:3
3       ldr sp, =stack_top
(gdb) l
1       .global reset
2       reset:
3       ldr sp, =stack_top
4       bl main
5       stop:
6       bl stop(gdb)
(gdb) Line number 7 out of range; startup.s has 6 lines.
display/3i $pc
1: x/3i $pc
=> 0x10000 <reset>:  ldr    sp, [pc, #4]    ; 0x1000c <stop+4>
0x10004 <reset+4>:  bl     0x10010 <main>
0x10008 <stop>:    bl     0x10008 <stop>
(gdb) b main
Breakpoint 1 at 0x10018: file app.c, line 9.
(gdb) b *0x10010
Breakpoint 2 at 0x10010: file app.c, line 6.
(gdb) |
```

1. **(b main)** breakpoint at main function (0x10018 first line of c code in main between 0x10010 & 0x10018 context switching and initializing stack of main).
2. **(b *0x10010)** breakpoint at address 0x10010 to watch context and initializing of main stack.

```
MINGW32:/e/Master/Embedded-System-Online-Diploma/Unit3_EmbeddedC/Unit3_EmbeddedC_lecture 3/Extended fo...
it3_EmbeddedC_lecture 3\Extended for lab1\learn-in-depth.elf...done.
(gdb) target remote localhost:1234
Remote debugging using localhost:1234
reset () at startup.s:3
3       ldr sp, =stack_top
(gdb) l
1       .global reset
2       reset:
3       ldr sp, =stack_top
4       bl main
5       stop:
6       bl stop(gdb)
(gdb) Line number 7 out of range; startup.s has 6 lines.
display/3i $pc
1: x/3i $pc
=> 0x10000 <reset>:    ldr    sp, [pc, #4]    ; 0x1000c <stop+4>
   0x10004 <reset+4>:  bl     0x10010 <main>
   0x10008 <stop>:    bl     0x10008 <stop>
(gdb) b main
Breakpoint 1 at 0x10018: file app.c, line 9.
(gdb) b *0x10010
Breakpoint 2 at 0x10010: file app.c, line 6.
(gdb) si
reset () at startup.s:4
4       bl main
1: x/3i $pc
=> 0x10004 <reset+4>:  bl     0x10010 <main>
   0x10008 <stop>:    bl     0x10008 <stop>
   0x1000c <stop+4>:  ldrdeq r1, [r1], -r12
(gdb) |
```

1. **(si → step instruction)** pc will point to next address 0x10004 instruction which will be fetched.
(s → step c code level one line code in c maybe equivalent to many assembly instruction so).

```
MINGW32:/e/Master/Embedded-System-Online-Diploma/Unit3_EmbeddedC/Unit3_EmbeddedC_lecture 3/Extended fo...
1: x/3i $pc
=> 0x10004 <reset+4>: b1      0x10010 <main>
   0x10008 <stop>:      b1      0x10008 <stop>
   0x1000c <stop+4>:    ldrdeq  r1, [r1], -r12
(gdb) c
Continuing.

Breakpoint 2, main () at app.c:6
6      {
1: x/3i $pc
=> 0x10010 <main>:      push    {r11, lr}
   0x10014 <main+4>:    add     r11, sp, #4
   0x10018 <main+8>:    ldr     r0, [pc, #4] ; 0x10024 <main+20>
(gdb) c
Continuing.

Breakpoint 1, main () at app.c:9
9      uart_send_string(stringBuffer);
1: x/3i $pc
=> 0x10018 <main+8>:    ldr     r0, [pc, #4] ; 0x10024 <main+20>
   0x1001c <main+12>:   b1      0x10028 <uart_send_string>
   0x10020 <main+16>:   pop     {r11, pc}
(gdb) l
4      uint8 stringBuffer[100] = "Learn-in-Depth: Mohamed Abdallah";
5      void main(void)
6      {
7          /* pass string to uart */
8          // name of array point to the address of first charater stringBuffer == &stringBuffer[0]
9          uart_send_string(stringBuffer);
10     }(gdb) |
```

1. (c → continue) will jump to first breakpoint at address 0x10010.
2. (c) again will jump to next breakpoint at address 0x10018 which first c code of main function at line 9.
3. (l show currently file app.c) we found that first line of c main at line 9 already.

```
MINGW32:/e/Master/Embedded-System-Online-Diploma/Unit3_EmbeddedC/Unit3_EmbeddedC_lecture 3/Extended fo...
Type "apropos word" to search for commands related to "word".
Command name abbreviations are allowed if unambiguous.
(gdb) print stringBuffer
$1 = "Learn-in-Depth: Mohamed Abdallah", '\000' <repeats 67 times>
(gdb) print stringBuffer[0]
(gdb) Undefined command: "". Try "help".
help
List of classes of commands:
aliases -- Aliases of other commands
breakpoints -- Making program stop at certain points
data -- Examining data
files -- Specifying and examining files
internals -- Maintenance commands
obscure -- Obscure features
running -- Running the program
stack -- Examining the stack
status -- Status inquiries
support -- Support facilities
tracepoints -- Tracing of program execution without stopping the program
user-defined -- User-defined commands

Type "help" followed by a class name for a list of commands in that class.
Type "help all" for the list of all commands.
Type "help" followed by command name for full documentation.
Type "apropos word" to search for commands related to "word".
Command name abbreviations are allowed if unambiguous.
(gdb) print stringBuffer[0]
$2 = 76 'L'
(gdb) |
```

1. (print stringBuffer[0]) print value of variable 76 ascii code.

```

files -- Specifying and examining files
internals -- Maintenance commands
obscure -- Obscure features
running -- Running the program
stack -- Examining the stack
status -- Status inquiries
support -- Support facilities
tracepoints -- Tracing of program execution without stopping the program
user-defined -- User-defined commands

Type "help" followed by a class name for a list of commands in that class.
Type "help all" for the list of all commands.
Type "help" followed by command name for full documentation.
Type "apropos word" to search for commands related to "word".
Command name abbreviations are allowed if unambiguous.
(gdb) display/3i $pc
2: x/3i $pc
=> 0x10038 <uart_send_string+16>:      b      0x10058 <uart_send_string+48>
    0x1003c <uart_send_string+20>:
        ldr r3, [pc, #48] ; 0x10074 <uart_send_string+76>
    0x10040 <uart_send_string+24>:      ldr      r2, [r11, #-8]
(gdb) where
#0  uart_send_string (
    p_tx_string=0x10078 <stringBuffer> "Learn-in-Depth: Mohamed Abdallah")
    at uart.c:9
#1  0x00010020 in main () at app.c:9
(gdb) |

```

- I set breakpoint at `uart_send_string` then pressed c to jump to it.
1. (where) currently at `uart_send_string` function in `uart.c` file and reached here by call was in `main` function in file `app.c` at line 9.

```

Mohamed@DESKTOP-AJUUKKD MINGW32 /e/Master/Embedded-System-Online-Diploma/Unit3_E
mbeddedC/Unit3_EmbeddedC_lecture 3/Extended for lab1 (main)
$ E:/Master/qemu/qemu-system-arm -M versatilepb -m 128M -s -S -nographic -kernel learn-in-depth.elf
Learn-in-Depth: Mohamed Abdallah

Mohamed@DESKTOP-AJUUKKD MINGW32 /e/Master/Embedded-System-Online-Diploma/Unit3_EmbeddedC/Unit3_EmbeddedC_lecture 3/Extended for lab1 (ma
in)
$ AC

Mohamed@DESKTOP-AJUUKKD MINGW32 /e/Master/Embedded-System-Online-Diploma/Unit3_EmbeddedC/Unit3_EmbeddedC_lecture 3/Extended for lab1 (ma
in)
$ E:/Master/qemu/qemu-system-arm -M versatilepb -m 128M -s -S -nographic -kernel learn-in-depth.elf
L|

```

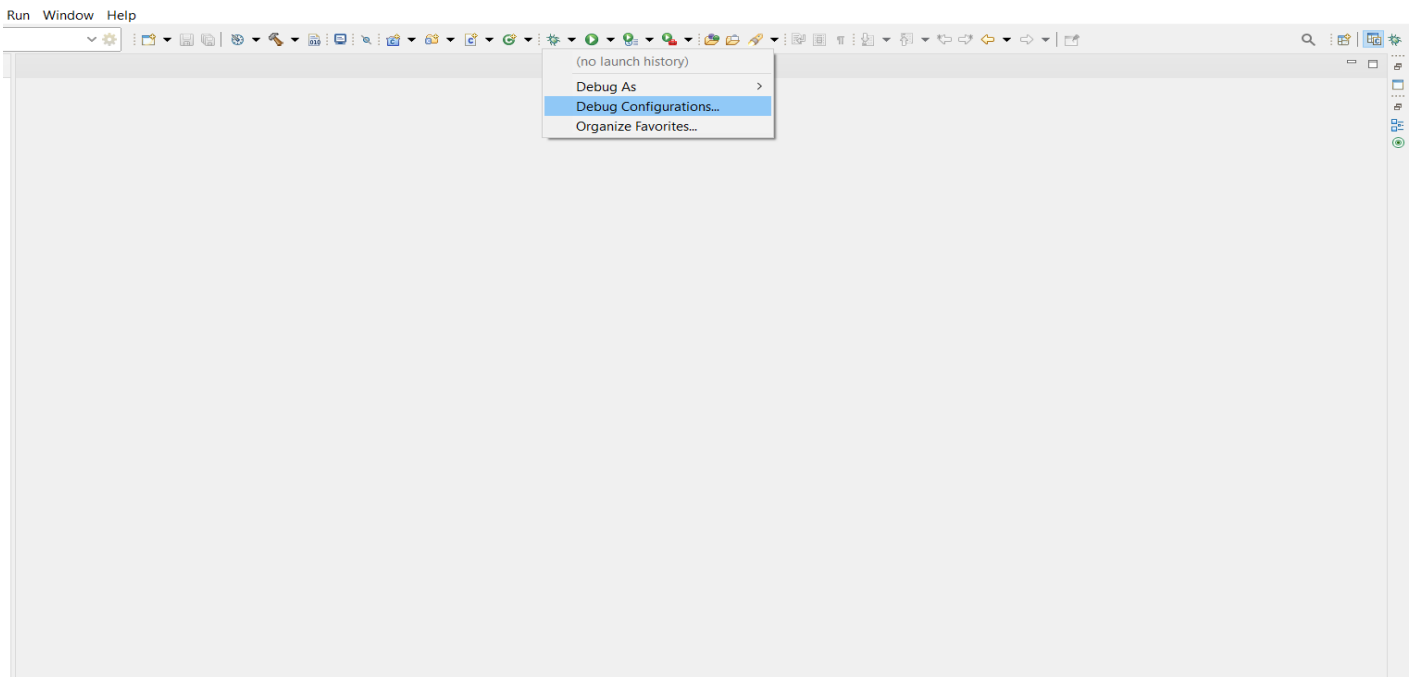
```

#1  0x00010020 in main () at app.c:9
(gdb) l
14         p_tx_string++;
15     }
16     }(gdb) s
12     UARTODR = (uint32_t)*p_tx_string;
2: x/3i $pc
=> 0x1003c <uart_send_string+20>:
        ldr r3, [pc, #48] ; 0x10074 <uart_send_string+76>
    0x10040 <uart_send_string+24>:      ldr      r2, [r11, #-8]
    0x10044 <uart_send_string+28>:      ldrb     r2, [r2]
1: x/3i $pc
=> 0x1003c <uart_send_string+20>:
        ldr r3, [pc, #48] ; 0x10074 <uart_send_string+76>
    0x10040 <uart_send_string+24>:      ldr      r2, [r11, #-8]
    0x10044 <uart_send_string+28>:      ldrb     r2, [r2]
(gdb) s
14         p_tx_string++;
2: x/3i $pc
=> 0x1004c <uart_send_string+36>:      ldr      r3, [r11, #-8]
    0x10050 <uart_send_string+40>:      add      r3, r3, #1
    0x10054 <uart_send_string+44>:      str      r3, [r11, #-8]
1: x/3i $pc
=> 0x1004c <uart_send_string+36>:      ldr      r3, [r11, #-8]
    0x10050 <uart_send_string+40>:      add      r3, r3, #1
    0x10054 <uart_send_string+44>:      str      r3, [r11, #-8]
(gdb) |

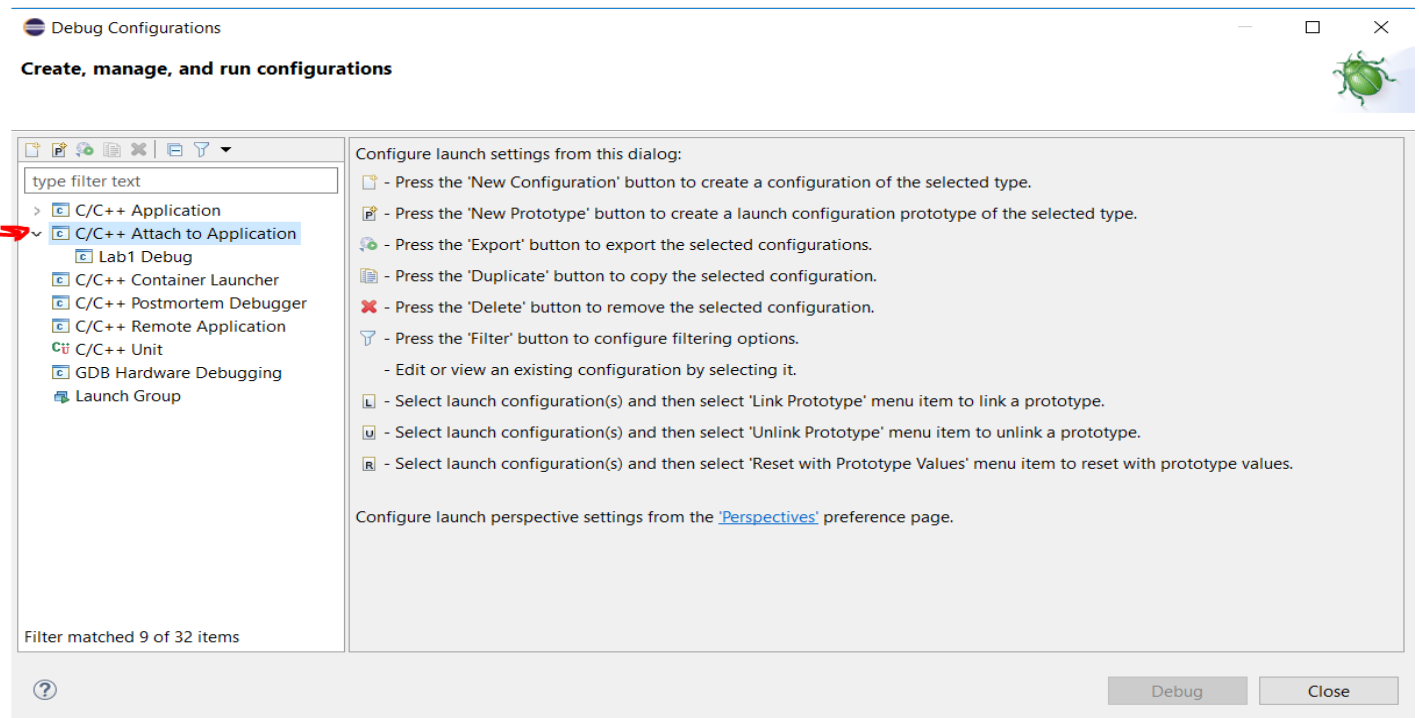
```

1. Using s will move line by line in c code to watch uart displaying our string written character by character.

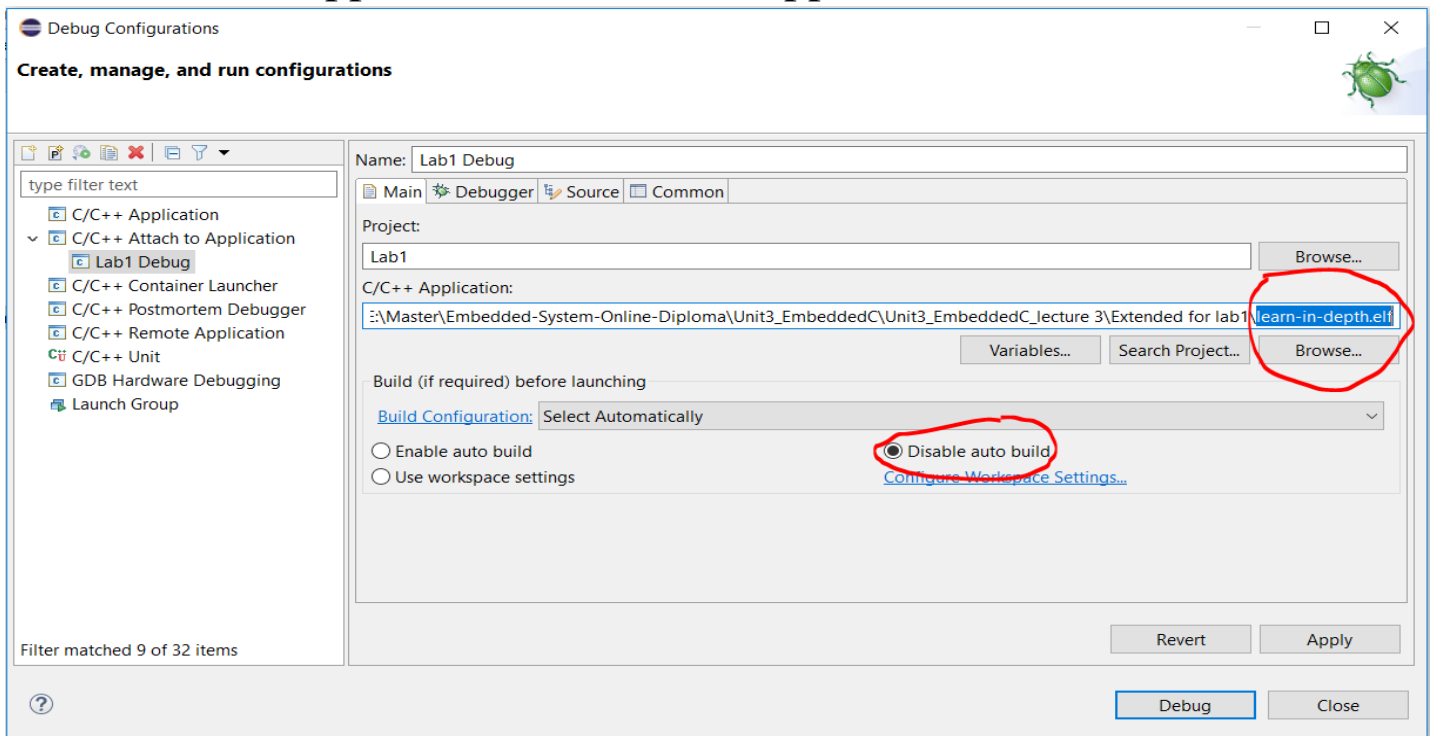
Debug using eclipse



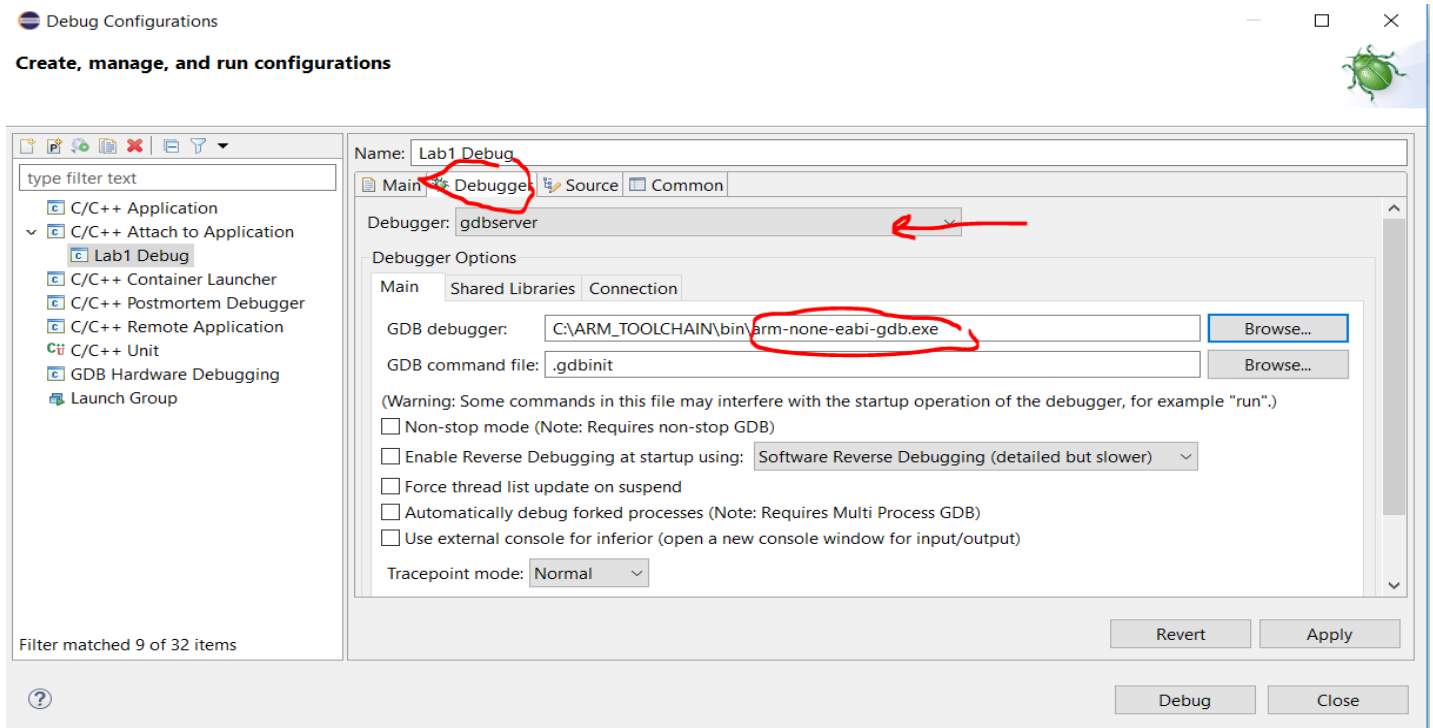
- Open empty project then select debug config.



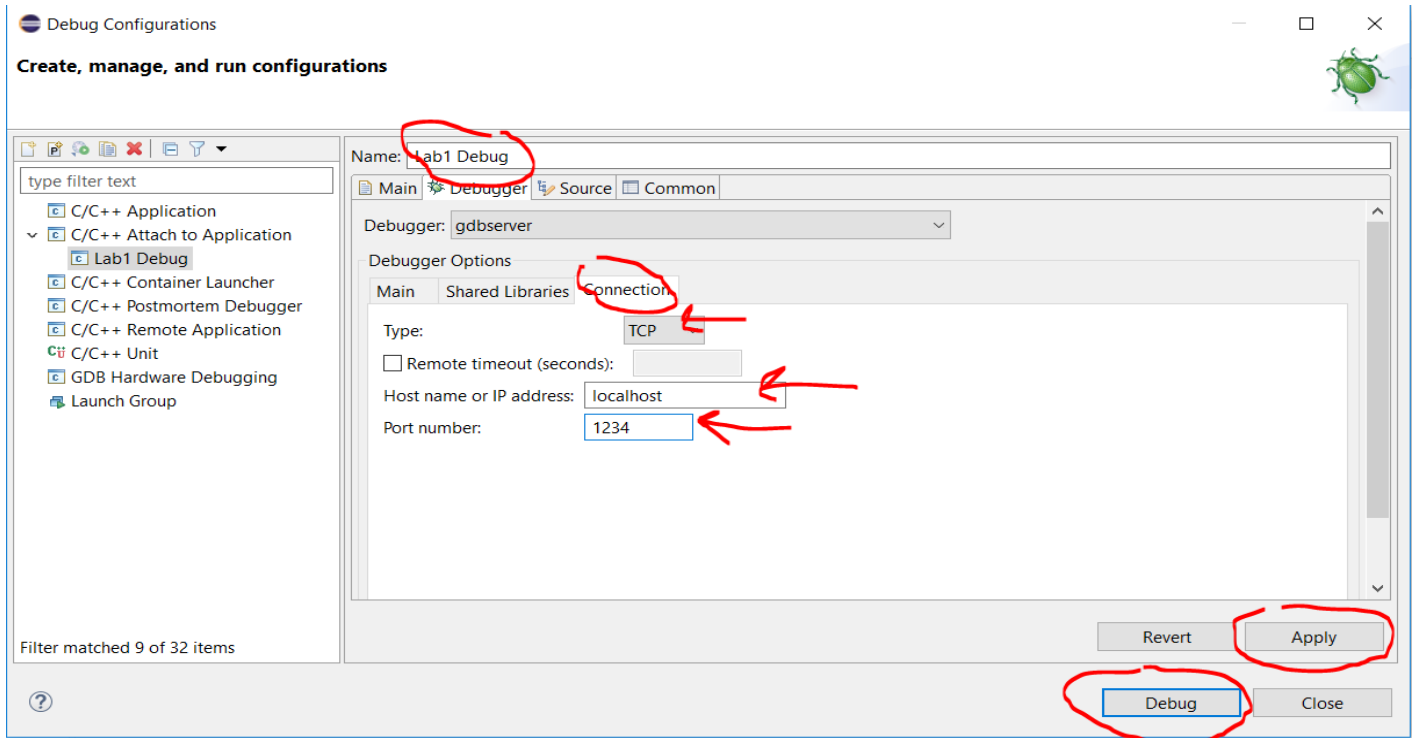
- Then dapple click on attach to application.



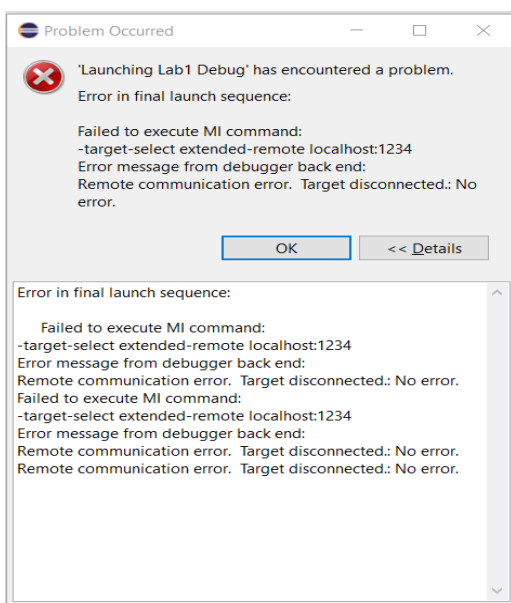
- Then select my .elf file and disable auto bulid.



- Then select debugger tab, select gdbserver then select gdb cross tool chain not native as our code will not run on our pc it will run on another machine.



- Select connection tab and select Type and set IP and port number.
- Give name to that debug config then apply and Debug.



- Maybe face this problem if you forgot to Open your board.

```
Console Registers Problems Executables Debugger Console x Memory
Lab1 Debug [C/C++ Attach to Application] C:\ARM_TOOLCHAIN\bin\arm-none-eabi-gdb.exe (7.5.1)
This GDB was configured as "--host=i686-pc-mingw32 --target=arm-none-eabi".
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
pwd
Working directory E:\Master\Embedded-System-Online-Diploma\unit2_C_Programming\codeforces\Lab1.
file learn-in-depth.elf
A program is being debugged already.
Are you sure you want to change the file? (y or n) [answered Y; input not from terminal]
Reading symbols from E:\Master\Embedded-System-Online-Diploma\unit2_C_Programming\codeforces\Lab1\learn-in-depth.elf...done.
```

- You need to know your current directory by using **pwd** and copy your .elf there then write this command **file file-name.elf**.
- You will see that gdb can now read symbols successfully.

The screenshot shows the ARM-none-eabi-gdb GUI with three main panels:

- Project Explorer:** Shows the file structure with 'Lab1 Debug [C/C++ Attach to Application]' and 'learn-in-depth.elf' selected.
- Disassembly Window:** Displays assembly code for 'reset' and 'main' functions. The 'reset' function includes instructions like 'ldn sp, =stack_top' and 'bl main'. The 'main' function includes instructions like 'ldr sp, [pc, #4]; 0x1000c' and 'bl 0x10008'.
- Console Window:** Shows the output of the 'pwd' command, the 'file learn-in-depth.elf' command, and the 'reset () at startup.s:4' command.

- Now he displayed current file and you can debug using GUI.

Makefile

The screenshot shows a development environment with three main components:

- File Explorer:** A list of files including `app.c`, `linker_script.ld`, `makefile`, `platform_Types.h`, `startup.s`, `uart.c`, and `uart.h`.
- Makefile:** A text editor showing the following content:

```
1  #@copyright : Mohamed Abdallah
2  startup.o: startup.s
3      arm-none-eabi-as.exe -mcpu=arm926ej-s -g startup.s -o startup.o
4  clean_all:
5      rm *.elf *.bin *.o
6  clean:
7      rm *.elf *.bin
```
- Terminal:** A terminal window showing the execution of `git commit` and `git config` commands. It then shows the command `make clean_all` being executed, which results in an error: `rm: cannot remove '*.elf': No such file or directory`. The error message is circled in red.

1. Clean_all → target, no dependences, role: remove all files .elf, .bin and .o.
2. How to call it? **make** then write **target name**, if we didn't write target it will execute first target.
3. startup.o → target, dependences → startup.s, role → assembler. Succeed

```
2/8/2022 10:58 Mohamed@DESKTOP-A2WKK10 /cygdrive/e/Master/Embedded-System-Online-Diploma/Unit3_EmbeddedC/Unit3_EmbeddedC_Lecture 3/Extended for lab1
$ make
arm-none-eabi-as.exe -mcpu=arm926ej-s -g startup.s -o startup.o
2/11/2022 2:58 startup.s: Assembler messages:
2/11/2022 2:58 startup.s: Warning: end of file not at end of a line; newline inserted
arm-none-eabi-gcc.exe -c -g -I . -mcpu=arm926ej-s uart.c -o uart.o
2/11/2022 2:58 arm-none-eabi-gcc.exe -c -g -I . -mcpu=arm926ej-s app.c -o app.o
2/6/2022 7:04 arm-none-eabi-ld.exe -T linker_script.ld uart.o app.o startup.o -o learn-in-depth.elf
2/6/2022 7:04 arm-none-eabi-objcopy.exe -O binary learn-in-depth.elf learn-in-depth.bin
2/11/2022 2:58 =====Build succeeded=====
2/6/2022 7:04
2/11/2022 2:58
2/6/2022 7:04
2/6/2022 7:04
2/6/2022 7:04
2/11/2022 2:58
```

```
1 #copyright : Mohamed Abdallah
2 all: learn-in-depth.bin
3 @echo =====Build succeeded=====
4 startup.o: startup.s
5 arm-none-eabi-as.exe -mcpu=arm926ej-s -g startup.s -o startup.o
6 uart.o: uart.c
7 arm-none-eabi-gcc.exe -c -g -I . -mcpu=arm926ej-s uart.c -o uart.o
8 app.o: app.c
9 arm-none-eabi-gcc.exe -c -g -I . -mcpu=arm926ej-s app.c -o app.o
10 learn-in-depth.elf: startup.o uart.o app.o
11 arm-none-eabi-ld.exe -T linker_script.ld uart.o app.o startup.o -o learn-in-depth.elf
12 learn-in-depth.bin: learn-in-depth.elf
13 arm-none-eabi-objcopy.exe -O binary learn-in-depth.elf learn-in-depth.bin
14 clean_all:
15 rm *.elf *.bin *.o
16 clean:
17 rm *.elf *.bin
```

1. target → startup.o, dependencies → startup.s, role → assembler.
2. Target → uart.o, dependencies → uart.c, role → compile but not link (-c).
3. Same as 2.
4. Target → learn-in-depth.elf, dependencies → startup.o, uart.o, app.o role → link.
5. Target → learn-in-depth.bin, dependencies → learn-in-depth.elf, role → obtain img to burn on board (without debugging info).
6. As when we call make, the first target will be executed we put all **target** and made it depended on .bin so it will generate all missing dependencies files.
7. Call make it generated all files and build succeeded as show.

- app.c
- app.o
- learn-in-depth.bin
- learn-in-depth.elf
- linker_script.ld
- makefile
- platform_Types.h
- startup.o
- startup.s
- uart.c
- uart.h
- uart.o

```
2/11/2022 3:05 PM $ make
arm-none-eabi-gcc.exe -c -g -I . -mcpu=arm926ej-s app.c -o app.o
arm-none-eabi-ld.exe -T linker_script.ld uart.o app.o startup.o -o learn-in-depth.elf
arm-none-eabi-objcopy.exe -O binary learn-in-depth.elf learn-in-depth.bin
=====Build succeeded=====
2/11/2022 3:05 PM

2/6/2022 7:06 PM E:\Master\Embedded-System-Online-Diploma\Unit3_EmbeddedC\Unit3_EmbeddedC_lecture 3\Extended for lab1\app.c - Sublime Text (UNREGISTERED)
2/11/2022 2:49 PM File Edit Selection Find View Goto Tools Project Preferences Help
2/6/2022 7:06 PM uart.c x makefile x download playlist.txt x app.c x uart.h x startup.s x
2/11/2022 2:50 PM 1 #include "uart.h"
2/6/2022 7:06 PM 2 #include "platform_types.h"
2/6/2022 7:06 PM 3 /* global array of char act as string holding our string which we need it to be written on uart */
2/6/2022 7:06 PM 4 uint8 stringBuffer[100] = "Learn-in-Depth: Mohamed Abdallah";
2/11/2022 2:50 PM 5 uint8 stringBuffer2[100] = "Learn-in-Depth: Mohamed Abdallah";
6 void main(void)
7 {
8     /* pass string to uart */
9     // name of array point to the address of first charater stringBuffer == &stringBuffer[0]
10    uart_send_string(stringBuffer);
11 }
```

1. We will make some changes on app.c file as shown.
2. Then build again we notice that it build app.c only then link again and generate binary image finally.
(changed files only will be build and all files depended on that changed files as shown not all file).

```

1  #@copyright : Mohamed Abdallah
2  CC=arm-none-eabi-
3  CFLAGS= -g -mcpu=arm926ej-s
4  INCS=-I .
5  LIBS=
6  SRC=$(wildcard *.c)
7  OBJ=$(SRC:.c=.o)
8  AS=$(wildcard *.s)
9  ASOBJ=$(AS:.s=.o)
10 PROJECTNAME=learn-in-depth
11 all: $(PROJECTNAME).bin
12     @echo =====Build succeeded=====
13
14 %.o: %.s
15     $(CC).as.exe $(CFLAGS) $< -o $@
16
17 %.o: %.c
18     $(CC).gcc.exe -c $(CFLAGS) $(INCS) $< -o $@
19
20 $(PROJECTNAME).elf: startup.o uart.o app.o
21     $(CC).ld.exe -T linker_script.ld $(LIBS) $(OBJ) $(ASOBJ) -o $@
22
23 $(PROJECTNAME).bin: $(PROJECTNAME).elf
24     $(CC).objcopy.exe -O binary $< $@
25
26 clean_all:
27     rm *.elf *.bin *.o
28
29 clean:
30     rm *.elf *.bin

```

```

/cygdrive/e/Master/Embedded-System-Online-Diploma/Unit3_EmbeddedC/Unit3_EmbeddedC_lecture 3/Extended for lab1
=====Build succeeded=====
arm-none-eabi-objcopy.exe -O binary learn-in-depth.elf learn-in-depth.bin
Mohamed@DESKTOP-AJUUKKD /cygdrive/e/Master/Embedded-System-Online-Diploma/Unit3_EmbeddedC/Unit3_EmbeddedC_lecture 3/Extended for lab1
$ make clean_all
rm *.elf *.bin *.o
Mohamed@DESKTOP-AJUUKKD /cygdrive/e/Master/Embedded-System-Online-Diploma/Unit3_EmbeddedC/Unit3_EmbeddedC_lecture 3/Extended for lab1
$ make
arm-none-eabi-as.exe -g -mcpu=arm926ej-s startup.s -o startup.o
startup.s: Assembler messages:
startup.s: Warning: end of file not at end of a line; newline inserted
arm-none-eabi-gcc.exe -c -g -mcpu=arm926ej-s -I . uart.c -o uart.o
arm-none-eabi-gcc.exe -c -g -mcpu=arm926ej-s -I . app.c -o app.o
arm-none-eabi-ld.exe -T linker_script.ld app.o uart.o startup.o -o learn-in-depth.elf
arm-none-eabi-objcopy.exe -O binary learn-in-depth.elf learn-in-depth.bin
=====Build succeeded=====
Mohamed@DESKTOP-AJUUKKD /cygdrive/e/Master/Embedded-System-Online-Diploma/Unit3_EmbeddedC/Unit3_EmbeddedC_lecture 3/Extended for lab1
$

```

1. `%.s` → anyfile.s.
2. `$<` → replaced with dependences file.
3. `$@` → replaced with target file.
4. `$(INC)` → replaced with value of the variable.
5. Wildcard `*.c` → all files.c.
6. `SRC:.c=.o` → replace .c with .o in files of SRC.